Information Retrieval Exercises

Assignment 3:

Boolean Information Retrieval with Lucene

Samuele Garda(gardasam@informatik.hu-berlin.de)
Lucene

- Java-based search engine
  - Apache Open Source Project
  - Widespread library for full text search
  - Related projects: ElasticSearch, Solr, Tika, Nutch, ...

- We will use the core library of Lucene!

- Requires two steps:
  - Indexing: Create a Lucene index with the documents
  - Searching: Parse query to search the index
Task

- Same as previous assignment:
  - Build index/indices from unstructured text
  - Solve queries with index/indices

- But this time use **Lucene (v8.8.2):**
  - [https://lucene.apache.org/core/8_8_2/index.html](https://lucene.apache.org/core/8_8_2/index.html)

- with its **Query Parser:**
Lucene: Basic concepts

- Lucene builds inverted indices and allows queries on these indices

- A Document is the unit of search and index
  - A document consists of one or more fields
  - A field is a key-value pair

- Indexing involves adding documents to an IndexWriter

- Searching involves retrieving documents via an IndexSearcher
Lucene: Basic concepts

• Tokenizers: break field data into lexical units, or tokens

• Filters: examine a stream of tokens and keep, transform or discard them

• Analyzers = tokenizers + filters
  – The output of an analyzer is used to query or build indices

• Use the same analyzer for querying and building indices!
Lucene: Analyzers

• Multiple default tokenizers, i.e.:
  - LetterTokenizer: divide text at non-characters
  - WhiteSpaceTokenizer: divide text at whitespace characters
  - StandardTokenizer: grammar-based tokenizer

• Multiple default filters, i.e.:
  - LowerCaseFilter: converts any uppercase letters to lowercase
  - Word Stemming filters (Kstem, Hunspell, Snowball Porter, …)

• Multiple default analyzers, i.e.:
  - SimpleAnalyzer: LetterTokenizer, LowerCaseFilter
  - StandardAnalyzer: StandardTokenizer, LowerCaseFilter, English stop words
  - WhiteSpaceAnalyzer: WhiteSpaceTokenizer
  - StopAnalyzer: LetterTokenizer, LowerCaseFilter, English stop words
Lucene API: Field types

• Fields types for text:
  – TextFields will be tokenized. Used for texts that needs to be tokenized
  – StringFields will be treated as a single term. Used for atomic values that are not to be tokenized

• Many other typed fields:
  – IntPoint/LongPoint: int/long indexed for exact/range queries
  – FloatPoint/DoublePoint: float/double indexed for exact/range queries

• Field.Store.YES : indexed & returned as result
• Field.Store.NO : indexed but not returned as result
Lucene Query Parser syntax

• You have to support the Query Parser syntax:
  – term query:
    
    plot:never

  – phrase query
    
    plot:"Luke Skywalker"

  – AND + OR query
    
    (plot:Innocent OR plot:Guilty) AND plot:crime

  – Wildecard query
    
    plot:unfaithful* AND plot:husband
Lucene Query Parser syntax

- Fuzzy Searches (optional maximum number of edits allowed)
  
  `plot:men~1 AND plot:women~1`

- Range Searches
  

- Prohibit operator
  
  `plot:Marvel -plot:DC`

- There is a built-in Query Parser for this in Lucene:
  
  - with more features: NOT queries, proximity, regular expressions, ...
Revisted: Movie corpora

• Corpus from previous assignment:
  – Plain text, ~400 MB

• You can reuse your file parser!

• Supported document types and their syntax in the corpus:
  – movie: MV: <title> (<year>)
  – series: MV: "<title>" (<year>)
  – episode: MV: "<title>" (<year>) {<episodetitle>}
  – television: MV: <title> (<year>) (TV)
  – video: MV: <title> (<year>) (V)
  – videogame: MV: <title> (<year>) (VG)
Corpus preprocessing

- Convert all words to lower case (case-insensitive search and indexing)

- Use:
  - word tokenization (both for term and phrase search)
  - stop word removal, but
  - no stemming

- There are built-in “Analyzers” for this in Lucene
Searchable fields

• Searchable fields are as follows:
  - title
  - plot (if a document has multiple plot descriptions they can be appended)
  - type (movie, series, episode, television, video, videogame; see next slide)
  - year (optional)
  - episodetitle (optional, only for episodes)

• There is a built-in MultiFieldQueryParser for this in Lucene!
ENOUGH BLABLA!!

JUST SHOW ME THE CODE!!
Getting started

• Get Apache Lucene v8.8.2 core and queryparser library
  
  - **Direct download:**
  
  - **Maven:**
    
    ```xml
    <dependency>
      <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-queryparser</artifactId>
      <version>8.8.2</version>
    </dependency>
    
    <dependency>
      <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-core</artifactId>
      <version>8.8.2</version>
    </dependency>
    
    <dependency>
      <groupId>org.apache.lucene</groupId>
      <artifactId>lucene-analyzers-common</artifactId>
      <version>8.8.2</version>
    </dependency>
    ```

    ```xml
    <dependencies>
      ...
    </dependencies>
    ```
Getting started

```xml
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.2.4</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
  <transformers>
    <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">
      <resource>META-INF/services/org.apache.lucene.codecs.Codec</resource>
    </transformer>
    <transformer implementation="org.apache.maven.plugins.shade.resource.AppendingTransformer">
      <resource>META-INF/services/org.apache.lucene.codecs.PostingsFormat</resource>
    </transformer>
    <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
      <mainClass>BooleanQueryLucene</mainClass>
    </transformer>
  </transformers>
</build>
```
Lucene API: Indexing

• Specify the analyzer to use
  Analyzer myAnalyzer = new StandardAnalyzer(); // or another Analyzer!

• Specify a directory and an index writer
  Directory index = FSDirectory.open(Paths.get(directory));
  IndexWriterConfig config = new IndexWriterConfig(myAnalyzer);
  IndexWriter writer = new IndexWriter(index, config);

• Create a document and add this document to the index:
  Document doc = new Document();
  doc.add(new StringField("id", id, Field.Store.YES));
  doc.add(new TextField("title", title, Field.Store.YES));
  writer.addDocument(doc);

• Close index writer:
  writer.commit()
  writer.close();
Lucene API: Querying

• Open Lucene index for searching
  ```java
  IndexReader indexReader = DirectoryReader.open(index);
  IndexSearcher indexSearcher = new IndexSearcher(indexReader);
  ```

• Parse title:<querystr> using the analyzer
  ```java
  Query query = new QueryParser("title", myAnalyzer).parse(querystr);
  ```

• Retrieve all results
  ```java
  TopDocs hits = indexSearcher.search(query, Integer.MAX_VALUE);
  long totalHits = hits.totalHits;
  for (ScoreDoc result: hits.scoreDocs) {
    Document document = indexReader.document(result.doc);
  }
  ```
```java
public class BooleanQueryLucene {

    /**
     * DO NOT CHANGE THE CONSTRUCTOR. DO NOT ADD PARAMETERS TO THE CONSTRUCTOR.
     */
    public BooleanQueryLucene() {
    }

    /**
     * A method for reading the textual movie plot file and building a Lucene index.
     * The purpose of the index is to speed up subsequent boolean searches using
     * the {link #booleanQuery(String) booleanQuery} method.
     * <p>
     * DO NOT CHANGE THIS METHOD'S INTERFACE.
     * <p>
     * @param plotFile the textual movie plot file 'plot.list', obtainable from <a
     * href="http://www.imdb.com/interfaces"&gt;http://www.imdb.com/interfaces</a> for personal, non-commercial
     * use.
     */
    public void buildIndices(Path plotFile) {
        // TODO: insert code here
    }

    /**
     * A method for performing a boolean search on a textual movie plot file after
     * Lucene indices were built using the {link #buildIndices(String) buildIndices}
     * method.
     * DO NOT CHANGE THIS METHOD'S INTERFACE.
     * <p>
     * @param queryString the query string, formatted according to the Lucene query syntax.
     * @return the exact content (in the textual movie plot file) of the title
     * lines (starting with "MV: ") of the documents matching the query
     */
    public Set<String> booleanQuery(String queryString) {
        // TODO: insert code here
        return new HashSet<>();
    }

    /**
     * A method for closing any resources (e.g. open file handles or a thread pool).
     * <p>
     * DO NOT CHANGE THIS METHOD'S INTERFACE.
     */
    public void close() {
        // TODO: you may insert code here
    }
}
```
Submission

• We provide you with:
  − queries_lucene.txt: file containing exemplary queries
  − results_lucene.txt: file containing the expected results of running these queries
  − a main method for testing your code (which expects as parameters the corpus file, the queries file and the results file)
Submission

• Make sure that you...
  - ... did not change or remove any code from BooleanQueryLucene.java
  - ... did not alter the functions’ signatures (types of parameters, return values)
  - ... only use the default constructor and don’t change its parameters
  - ... did not change the class or package name
  - ... named your jar BooleanQueryLucene.jar
Submission

• Test your jar before submitting by running the examples queries on gruenau
  - java -jar BooleanQueryLucene.jar <plot list file> <queries file> <results file>
  - You might have to increase the JVM‘s heap size (e.g., -Xmx8g)
  - Your jar must run and answer all test queries correctly!

• Your program has to correctly answer all example queries correctly to pass the assignment!
Competition

• Index as fast as possible

• Note that everybody uses the same indexer (Lucene)

• Look for possible optimizations, e.g.

• Stay under 50 GB memory usage
Submissions

- Assignment 3 submission deadline:
  - **Group 1**: Tuesday, 15.06., 23:59 (midnight)
  - **Group 2**: Wednesday, 16.06., 23:59 (midnight)

- Presentations of the solutions for assignment 3
  - **Group 1**: Monday, 22.06.
  - **Group 2**: Wednesday, 23.06
Questions?